Lab 8

Exercise 1: Write a base class `Fraction` and a derived class `RFraction` that inherits `Fraction`. In the base class `Fraction`, use `int` variables `numer` and `denom` to represent the numerator (分子) and the denominator (分母) of the representing fractional number. `RFraction`, on the other hand represents `Fraction` in its reduced form. Provide constructors and proper IO operator overloading for the classes.

*You cannot define any member data in `RFraction`.

*The `operator>>` and `operator<<` should be only defined for `Fraction`.

Use the following client code and sample runs to test your program.

```
#include "Fraction.h"
#include "RFraction.h"
#include <iostream>
using namespace std;

int main(){
    Fraction f1(2, 4);
    Fraction f2;
    cout << "Enter a fraction: ";
    cin >> f2;
    cout << "Fractions are: ";
    cout << f1 << ", ";
    cout << f2 << endl;

    RFraction rf1(2, 4);
    RFraction rf2;
    cout << "Enter a fraction to be reduced: ";
    cin >> rf2;
    cout << "Reduced fractions are: ";
    cout << rf1 << ", ";
    cout << rf2 << endl;

    Fraction* rfp1 = new RFraction(-6, 16);
    Fraction* rfp2 = new RFraction();
    cout << "Enter a fraction to be reduced: ";
    cin >> *rfp2;
    cout << "Reduced fractions are: ";
    cout << *rfp1 << ", ";
    cout << *rfp2 << endl;
    delete rfp1;
    delete rfp2;
}
```

```
Enter a fraction: 4/12
Fractions are: 2/4, 4/12
Enter a fraction to be reduced: -4/12
Reduced fractions are: 1/2, -1/3
Enter a fraction to be reduced: -3/24
Reduced fractions are: -3/8, -1/8
```

Exercise 2: Based on the `DoubleArrayID.h` and `DoubleArray.h` files in the lecture note, practice implementing the `DoubleArrayID.cpp` and `DoubleArrayID.cpp`.

Use the following main to test your implementation:

```
#include "DoubleArray.h"
#include "DoubleArrayID.h"
#include <iostream>

using namespace std;

int main(){
    DoubleArrayID dar0;
    DoubleArrayID dar1(3, 0.1);
    DoubleArrayID dar2(6, 0.5);
    DoubleArray* dar3 = new DoubleArrayID(dar1);
    DoubleArrayID dar4;
    dar4 = dar2;
    cout << dar0 << dar1 << dar2 << *dar3 << dar4;

    delete dar3;

    return 0;
}
```

The output should be:

```
array size: 0, { }
:: { }
array size: 3, {0.1, 0.1, 0.1}
:: {8, 17, 54}
array size: 6, {0.5, 0.5, 0.5, 0.5, 0.5, 0.5}
:: {8, 17, 54, 49, 46, 95}
array size: 3, {0.1, 0.1, 0.1}
:: {8, 17, 54}
array size: 6, {0.5, 0.5, 0.5, 0.5, 0.5, 0.5}
:: {8, 17, 54, 49, 46, 95}
        delete[] in subclass destructor
delete[] in destructor!
        delete[] in subclass destructor
delete[] in destructor!
        delete[] in subclass destructor
delete[] in destructor!
        delete[] in subclass destructor
delete[] in destructor!
```

The answer is in the lecture note.